Implementation of skeleton frameworks

a) template based implementation

b) macro data flow based implementation

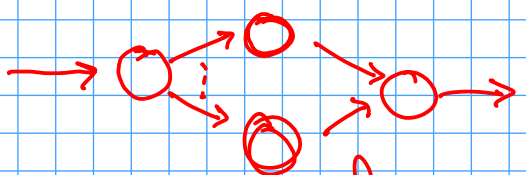
Prg = pipe(seq(..), pipe(seq(), seq()), map(seq()))

TEMPLATE BASED IMPLEMENTATION

TEMPLATE = { parametric process network implementing a pattern on a given target HW associated with a performance model

process = concurrent activity (Unit of Execution)

parametric = diff components or diff par. degree as parameters

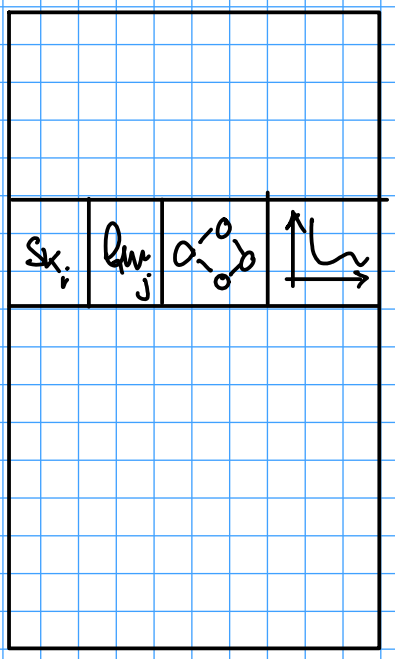


nw, nw' = f [f, g, h, ..]

implements: farm, map, stencil

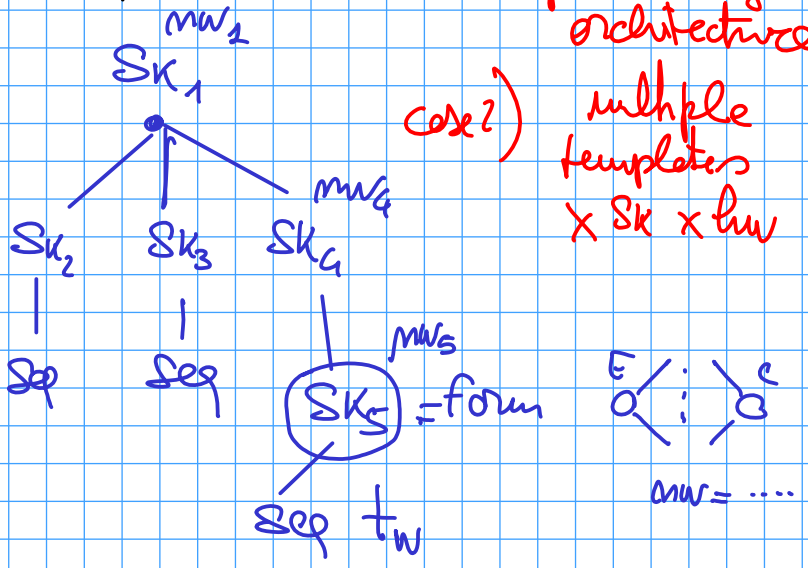
Perf(nw, t_w, ...) =

template library



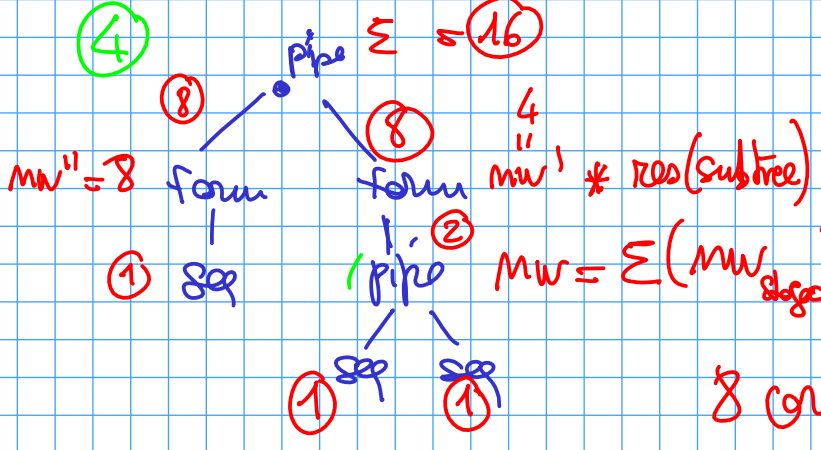
Case 1) 1 template $\forall$ Sk for the target architecture

Case 2) multiple templates $\times$ Sk $\times$ hw



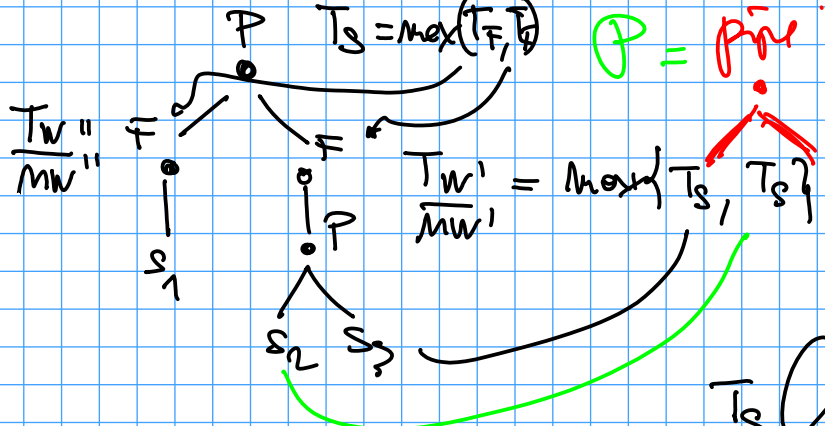
Step 1: visit the tree bottom up
$\forall$ node → assign the "best" template in lib
according to the performance model

Step 2: total count of resources needed
→ match the actual resources



template for
template pipe

8 cores available



P = pipe #Res

G = form #Res
mw = —

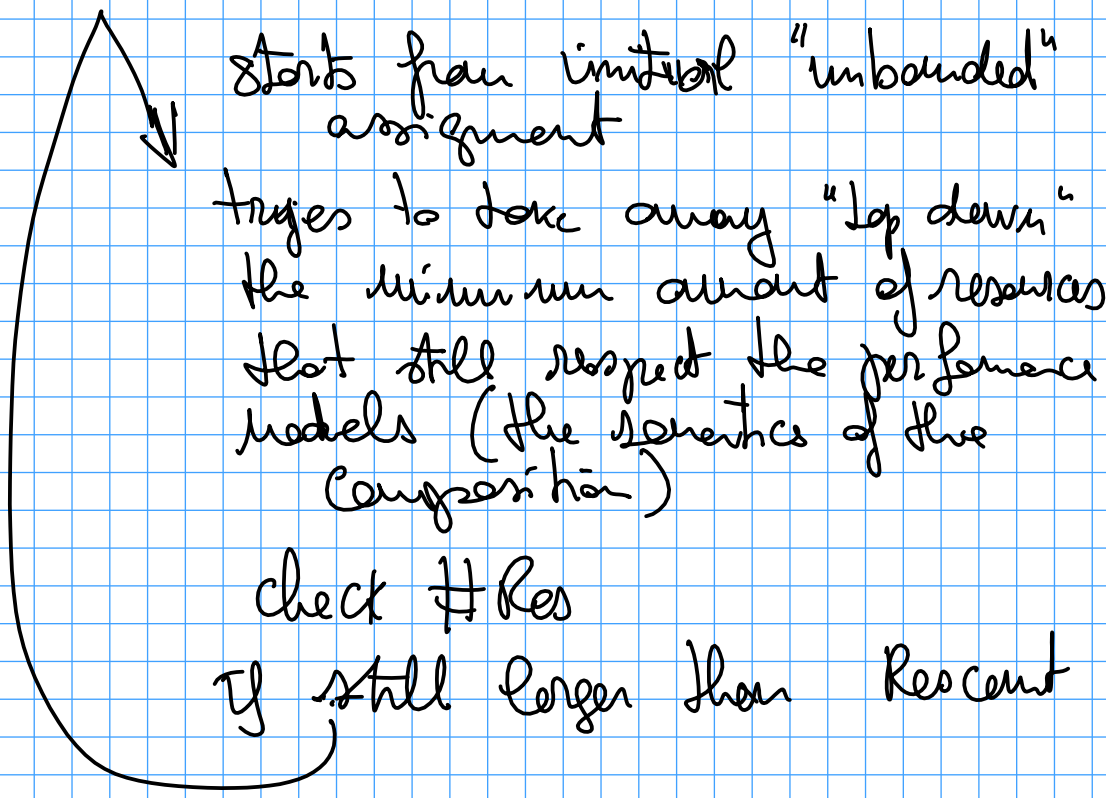
$$\#Res(\triangle) = mw'' + mw' \cdot 2$$

$$T_S(\triangle) = \max\left\{ \frac{T_{w''}}{mw''}, \frac{\max\{T_{S_2}, T_{S_3}\}}{mw'} \right\}$$

$\max T_S(\Delta)$ while $\#Res(\Delta) \leq Res_{cont}$

→ "exact" method

Iterative one :



Templates

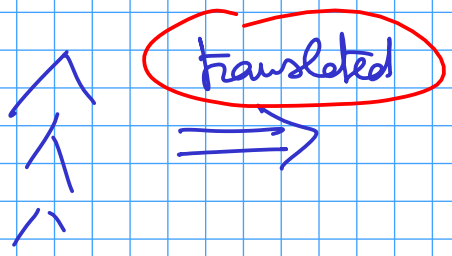
"static approach"

① build the template library (by experts of par exploitation techniques & target hw)

② set up a "compiler" (by experts ... ↙)

③ provide all this through a suitable syntax to the applications programmers

MACRO DATA FLOW APPROACH



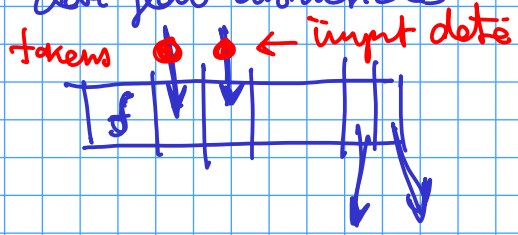
data flow graph

$(a+b)/(c-d)$

```
x = a+b;
y = c-d;
res = x/y;
```

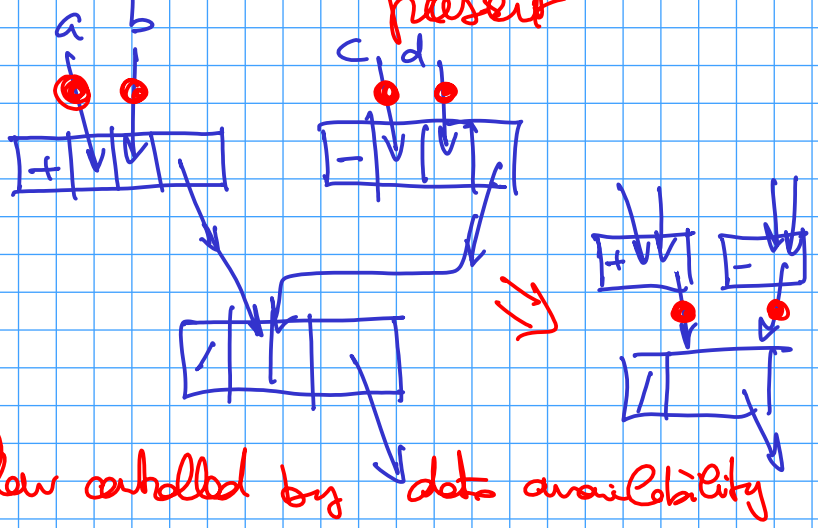
flow is controlled by Program Counter

data flow instruction



fireable if

MDF: may be computed only when all the input tokens are present

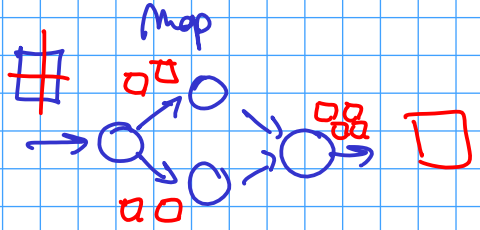


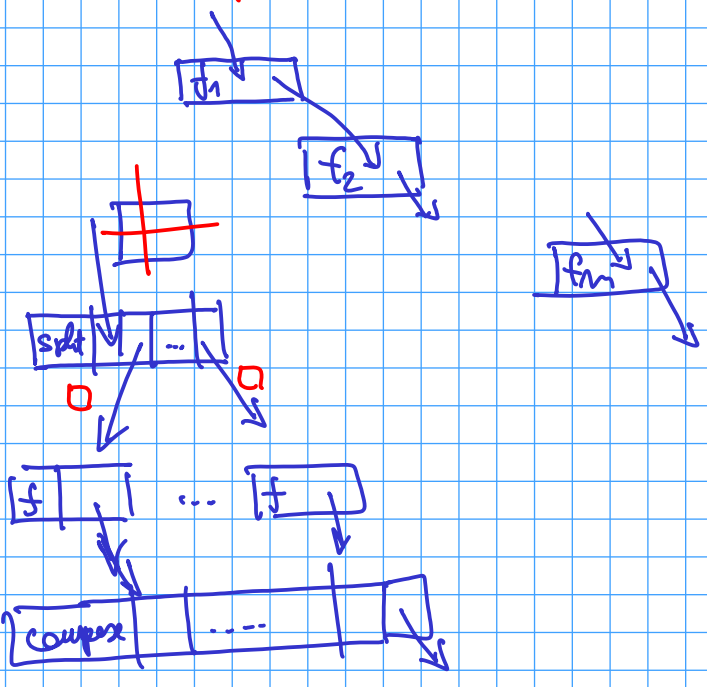
flow controlled by "data availability"

macro DF ⇒ "functions" computed by MDF: Could be very complex

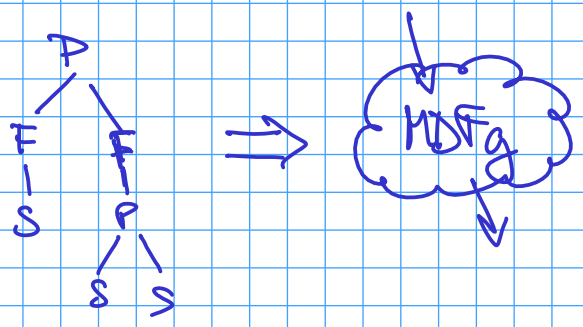
Pipeliné

Map

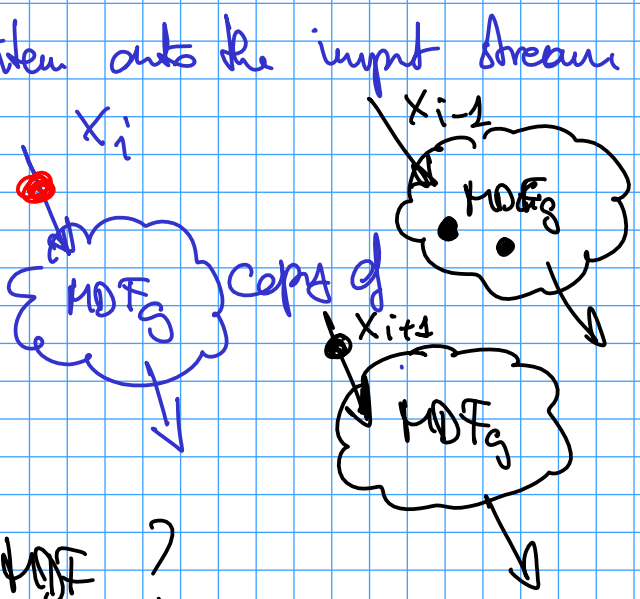




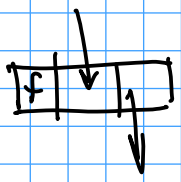
MDF : STREAMS



Token onto the input stream

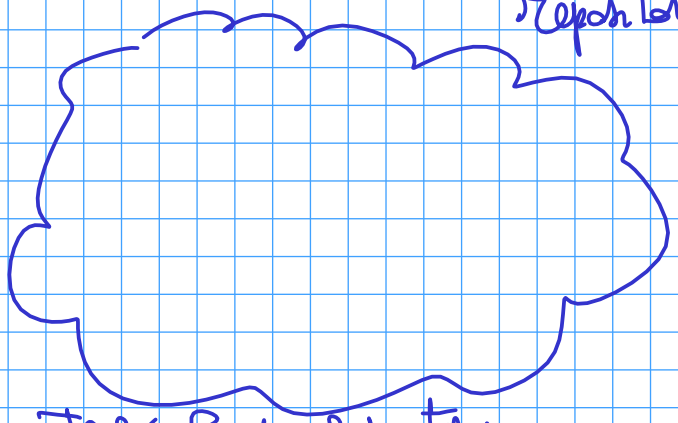


form (f) } how does it translate to MDF ?



translation (form (f)) = translation (f)

Implement a parallel MDF interpreter



repository of MDF graphs as instantiated by the stream mechanism

TASK Pool Repository

```
new EU :: cycle {
    read a fireable MDFi
    compute MDFi
    put back result on the MDFi
}
```

$\{MDF_i\} = \{ \text{fireable}(MDF_i) \} + \{ \text{not fireable}(MDF_i) \}$

MDF_i: